MLPNN实验报告

陈家豪 19307130210

一、数据处理

本次实验数据集为 Breast cancer dataset

通过sklearn.datasets获取数据

```
cancer = load_breast_cancer()
cancer_x=cancer.data
cancer_y=cancer.target
```

查看数据大小即部分数据：

```
print("加载完毕，数据大小：")
print(cancer_x.shape)
print(cancer_y.shape)
print("前5个数据：")
for i in range(5):
print(cancer_x[i],cancer_y[i])
```

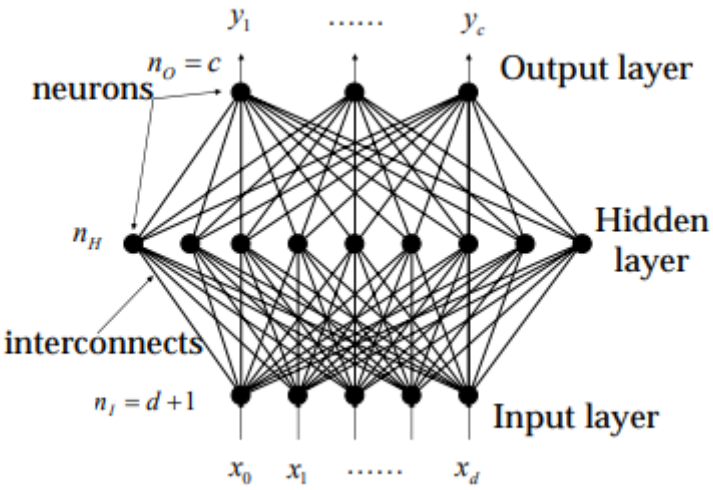
```
开始加载数据
加载完毕，数据大小：
(569, 30)
(569,)
前5个数据：
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01] 0
[2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
 7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
 5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
 2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
 2.750e-01 8.902e-02] 0
[1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
 1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
 6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
 2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
```

breast cancer 的影响因素有30个，结果为0,1，表示是否患病

对数据进行拆分, test_size=0.2：

```
x_train,x_test,y_train, y_test=train_test_split(cancer_x,cancer_y,test_size=0.2)
```

二、算法



如图为单层隐藏层的MLPNN模型示意图，输入层为数据集的数据，输出层为数据集的标签

隐藏层神经元表示为：

$$\text{net}_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} = \mathbf{w}_j^{\top} \mathbf{x}$$

每个单元发出它的激活函数：

$$y_j = g(\text{net}_j)$$

输出层为：

$$\text{net}_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = \mathbf{w}_k^{\top} \mathbf{y}$$

每个单元输出其激活函数：

$$y_k = f(\text{net}_k)$$

映射函数：

$$y_k = f\left(\underbrace{\sum_{j=0}^{n_H} f\left(\underbrace{\sum_{i=0}^{d} x_i w_{ji}}_{\text{net}_j} \right) w_{kj}}_{\text{net}_k} \right)$$

误差函数：

平方和误差函数：

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n)^2$$

三、模型

```
#'sgd' 指的是随机梯度下降；
clf = MLPClassifier(solver='sgd')
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
print(accuracy(y_test,y_pred))

#'lbfgs' 是准牛顿方法族的优化器；
clf2 = MLPClassifier(solver='lbfgs')
clf2.fit(x_train,y_train)
y_pred2 = clf2.predict(x_test)
print(accuracy(y_test,y_pred2))
```

模型 MLPClassifier, 其中参数solver表示权重优化的求解器。'lbfgs'是准牛顿方法族的优化器；'sgd'指的是随机梯度下降。

参数alpha是L2惩罚参数，默认值0.0001

参数max_iter是最大迭代次数，默认值200

参数hidden_layer_sizes是隐藏层神经元数，默认值（100，）

四、结果

```
PS H:\gitdemo\ML-HW\MLPNN> python MLPNN.py
start
正确率：
0.9210526315789473
0.9473684210526315
PS H:\gitdemo\ML-HW\MLPNN>
```

用'sgd'的正确率为：0.921

用'lbfgs'的正确率为：0.947